

Overview

The Handler Function is responsible for processing submitted inputs and generating the resulting output. When developing your Handler Function, you can do so locally on your PC or remotely on a RunPod GPU instance.

Example Handler Functions can be found within the repos of our runpod-workers.

Why use Handler Functions?

Handler Functions offer a paradigm shift in how you approach backend code execution:

- **Efficiency:** Focus on writing code that matters without worrying about the server lifecycle.
- **Cost-Effective:** You only pay for the time your functions are running, not idle server time.
- **Rapid Deployment:** Quickly update and deploy functions, enabling faster iteration and response to changes.

Your Handler Function only accepts requests using your own account's API key, not any RunPod API key.

Job input

Before we look at the Handler Function, it is essential first to understand what a job request input will look like; later, we will cover all of the input options in detail; for now, what is essential is that your handler should be expecting a JSON dictionary to be passed in. At a minimum, the input will be formatted as such:

```
{
  "id": "A_RANDOM_JOB_IDENTIFIER",
  "input": { "key": "value" }
}
```

Requirements

You will need to have the RunPod Python SDK installed; this can be done by running `pip install runpod`.

Basic Handler Function

```
# your_handler.py

import runpod # Required.

def handler(job):
    job_input = job["input"] # Access the input from the request.
    # Add your custom code here.
    return "Your job results"

runpod.serverless.start({"handler": handler}) # Required.
```

You must return something as output when your worker is done processing the job. This can directly be the output, or it can be links to cloud storage where the artifacts are saved. Keep in mind that the input and output payloads are limited to 2MB each

NOTE

Keep setup processes and functions outside of your handler function. For example, if you are running models make sure they are loaded into VRAM prior to calling `serverless.start` with your handler function.

Development and deployment

You should return something as output for when your Worker is done processing a job. This can be directly the output, or it can be links to cloud storage where the artifacts are saved.


Payloads are limited to:

- `run`: 10 MB.
- `runsync`: 20 MB.

If any errors are returned by the worker while running a `test_input` job, the worker will exit with a non-zero exit code. Otherwise, the worker will exit with a zero exit code. This can be used to check if the worker ran successfully, for example, in a CI/CD pipeline.

- For information on testing your handler locally, see [Local testing](#).

- For information on setting a continuous integration pipeline, see [Continuous integration](#).

 [Edit this page](#)